MARSV FOR PLAN 9. Kenji Okamoto (okamoto@granite.cias.osakafu-u.ac.jp), *Coll. Integrated Arts & Sciences, Osaka Prefecture Univ., Sakai, Osaka, 599-8531, Japan*, Yoshitatsu Suzuki (yoshitatsu.suzuki@ctc-g.co.jp), *now at Itochu Techno-Science Co., Tokyo*.

After GUI was spread, we have to deal with huge sized programs written for investigation of planetary data. There are many commercial based utilities to make those smaller such that GUI libraries or more abstracted forms aslanguage based IDL etc.. Those utilities are, unfortunately, suffered from very often updates, which makes us annoying sometime. We searched many operating systems and found Plan 9 from Bell Labs, which was newly designed from scrach with the aim of replacing Unix in a future, and has a very compact graphics system. Plan 9 is now open to researcheres without fee. Based on that Plan 9 OS, we expected to make a smaller sized and well featured GUI program for our investigation of Mars MOLA and image data, and got such with total C source codes of about 10,300 lines. We call the program as marsv.

Marsv has no icon, and only has top menu bar where some menues are shown as their default values. Each menu item has a list of other similer menu items behind it, which is shown by mouse button3. Mouse button2 executes the item shown in the menu bar, etc, which is similer to pulldown menu of X, but not same. Marsv displays each window as a stratified and tiled window as shown in the figure. The size of individual window can be changed by mouse at anytime and anywhere as if it has window manager like such in X.

Marsv reads many kinds of PDS formatted planetary images such as Viking, Voyager, Mars Global surveyer, Megellan etc. Mars MOLA grid data also can be read, and are laid over the corresponding Viking cube image processed by ISIS from USGS, and we can make contour map on the image. We can also measure interactively the height, longitude and latitude of individual mouse point in the image by "altitude" menu behind the "contour" top menu. When MOLA data alone are read, by specifying the wanted area by mouse, those height data are converted to brightness, and marsv shows them as an image, where botton1 reads the height, longitude and latitude of the point. Annotations can be attached by "pen" menu, and"pallet" menu changes the color of the couter map etc.. Marsv can also save the processed image to a postscript or Plan 9's PIC format file.

Marsv is designed for concurrent parallel programming using the thread(2) library of Plan 9, which means it has full scalability. Plan 9 has two different level graphic libraries of draw(2) and control(2), where the former is the lowest level graphic library, and may correspond to X toolkit library judged from its range of functionarity, and the latter is a higher level library which deals with some widget sets such as menu, button, text input etc. We used somewhat revised version of this control(2), because we needed 3 button functions for menu bar.

When marsv is running, we have three processes, one of which for watching the events (channel), and the other one accepting keyboard or mouse input, and the last contains threadmain() function of marsv program and many other threads called from the threadmain(). The last process is composed of many threads, each of which is called as a "task" or 'thread', and will be called when appropriate event has detected or more precisely any appropriate channel has the message to that thread. The event will be passed through channel, whic is a buffered or unbuffered queue for a fixed-size messages. In our case, one process has many tasks for real processing of image data, which can run only as one exclusive task in the process in a time interval we are concerning. In other words, only one task can run in the last process containing threadmain(). It is the most simple case study of thread library of Plan 9. Only the channel is the mean to communicate with each other for processes or tasks, and one of the rforked process will watch the channel list.

Those three processes are forked by use of light weight rfork(2) of Plan 9, and shares many of memory, and will not use so much memory space. In our case marsv uses 828 Kb memory area for the main process at its start. Those three processes uses preemptive time sharing, then, can run simultaneously, which enables rapid communication. It is the reason why those three preemptive time sharing processes are used for watching keyboard and mouse and channels.

We tried to design all the tasks simpler as possible as we can. However, we have to accept some nested tasks because of complexity of the processing the planetary data. The deepest nest level of marsv is seven, and seen when saving the processed image to a file. The sequence of this nesting is from top level thread of threadmain to imagerthread, canvasthread, viewerthread, savethread, filebrowsethread and selectthread from top to bottom.

From the programmer's point of view, one of the most difficult or cumbersome is coding to make all the windows size-flexible, because Plan 9 doesn't have window manager, but only has a function named resizecontrolset() in control(2) library the content of which must be defined by user. All the other windows also must be prepared for resizing by the user.

Marsv for Plan 9: Okamoto & Suzuki

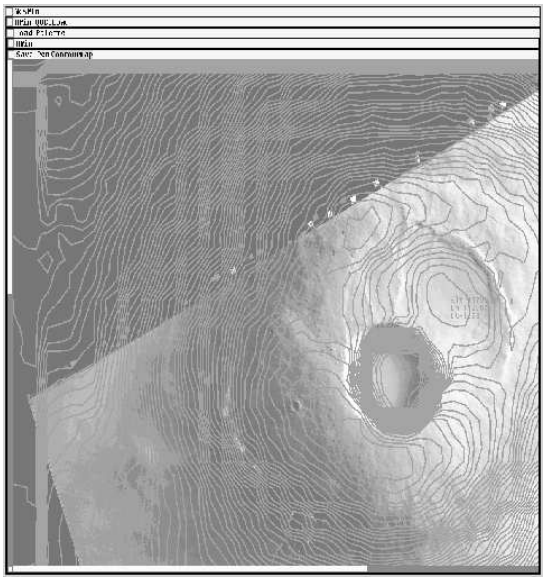

Figure 1: contour map

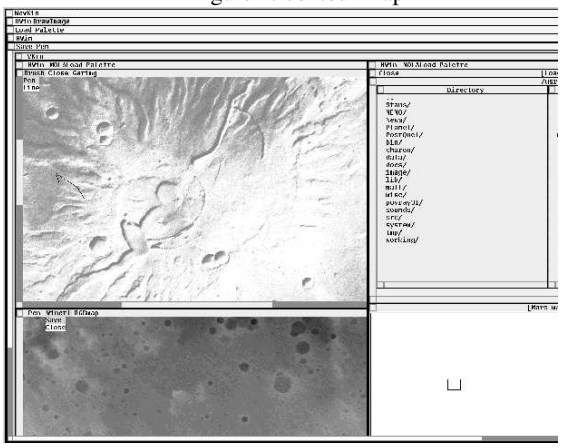

Figure 2: user interface